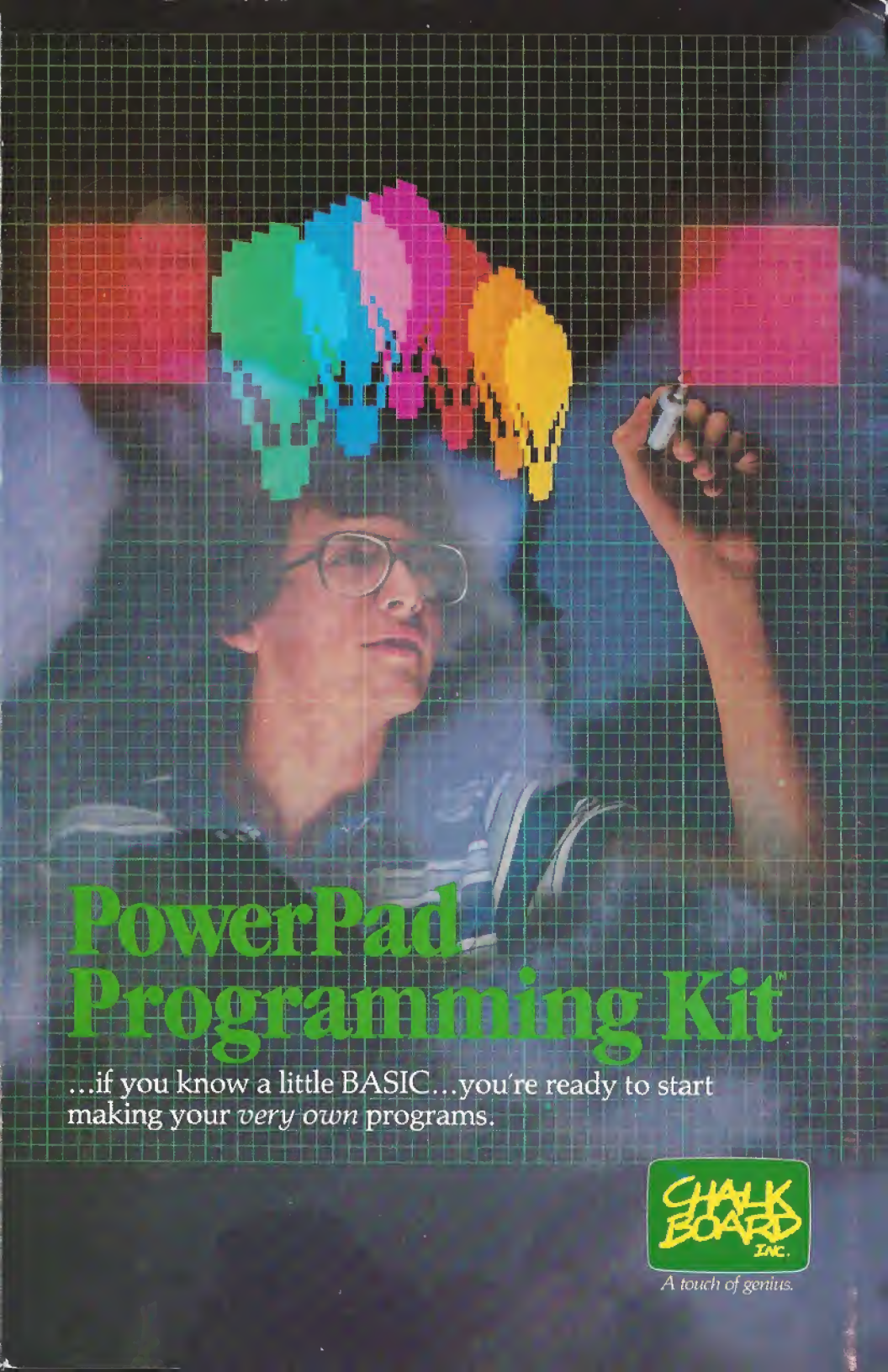


PowerPad Programming Kit™

...if you know a little BASIC...you're ready to start making your *very own* programs.

CHALK BOARD INC.

A touch of genius.

PowerPad Programming Kit™

User's Guide For The Atari® 400™/800™

Chalk Board, Inc.
3772 Pleasantdale Road
Atlanta, Georgia 30340
(404) 496-0101
(800) 241-3989 (Outside Georgia)

Table of Contents

Version 3.2/ 831030

Table of Figures

Figure

Table of Programs

Program

*In order to run the programs in this package you must have an Atari BASIC cartridge.

INTRODUCTION

The Chalk Board PowerPad offers an alternative to the typewriter-style keyboard in working with your home computer. For many people this development makes contact with the computer less frustrating and more fun.

But for those who have mastered the computer keyboard and are familiar with the BASIC programming language, PowerPad offers some exciting and powerful new dimensions to writing and executing programs. While you learn to write programs for PowerPad, you gain a deeper working knowledge of your own home computer.

The information in this manual explains and demonstrates to you how to give commands to PowerPad in Atari BASIC. It provides a series of activities which will familiarize you with PowerPad's programming. Lastly, there are some challenging questions for you to consider on your way to more sophisticated programming.

If you are unfamiliar with the operation of the Chalk Board PowerPad, be sure to refer to the PowerPad User's Guide, which accompanied your PowerPad. This guide contains instructions concerning how to connect the pad to your Atari home computer system. Find the section of the directions which illustrates use with the Atari computer.

Getting Started

- Do not try to begin without reading the PowerPad User's Guide.
- Refer to the section which illustrates the use of PowerPad with Atari home computers.
- Be sure to unplug your computer and TV **before** connecting PowerPad to your computer.
- Insert the overlay included with this product by gently guiding the overlay's frame into the groove around PowerPad's work surface.

PowerPad Test in BASIC

The Chalk Board PowerPad is a pressure-sensitive input device. It senses the touch of a finger (or any blunt stylus) and provides the location of the point of contact. The location point is given in the form of X and Y coordinates (similar to those found on the Cartesian Coordinate System). Figure 1 shows how the coordinates are arranged on the PowerPad surface. The values on each axis range from 0 to 119. There are 120 rows by 120 columns, or 14,400 different possible points which PowerPad can sense. Figure 1 also shows a sample point and the X and Y values that are returned by touching the pad at that point.

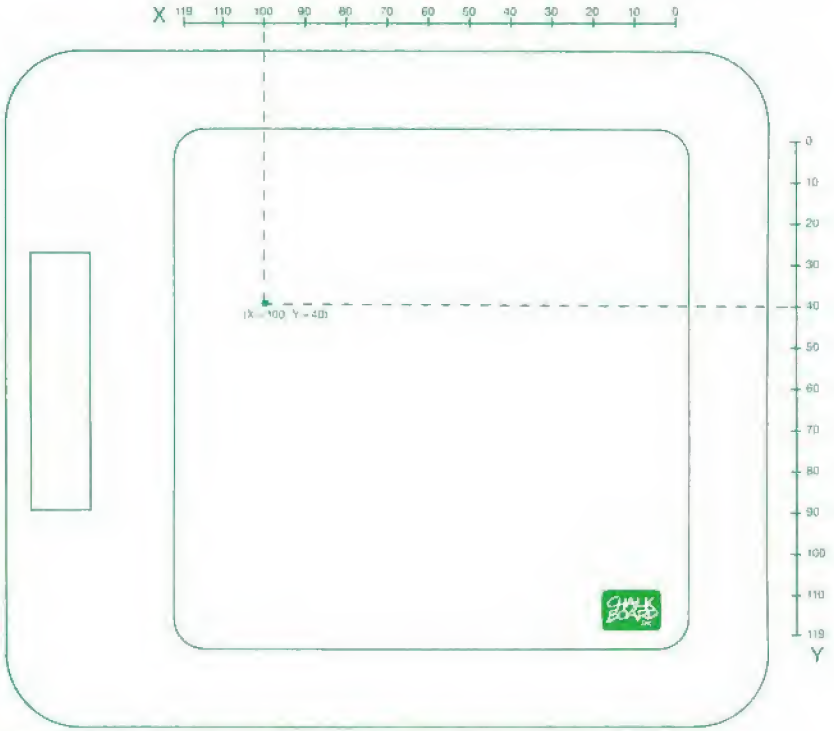


Figure 1. PowerPad Coordinate Arrangement and Sample Point

PowerPad sends the X and Y coordinates to your computer in a serial format. This means that the values for X and Y are converted to binary numbers and are sent one bit at a time. To read the X and Y values, the computer must read the first bit, request that the next bit be sent, read that bit, and so on, until all of the bits have been read. The computer must then reassemble those bits.

There are a total of 16 bits. The first two have no meaning and should be ignored. The following seven are for the Y value, and the last seven are for the X value.

Program 1, PowerPad Test, uses subroutines to scan the pad and print to the screen the X and Y coordinates of the point of contact. The subroutines can be called by any BASIC program to communicate with PowerPad.

The first subroutine (lines 10000–10050) initializes the variables used by the other subroutines.

The second subroutine (lines 11000-11130) sets up joystick port 1 to talk to PowerPad and tells PowerPad to begin scanning for points being touched. The fifth time that this subroutine is called, the message "UNABLE TO TALK TO POWERPAD. DID YOU PLUG PAD INTO JOYSTICK PORT 1?" is printed and the program ends.

The third subroutine (lines 12000-12100) determines whether PowerPad has been touched. If PowerPad has been touched, it is ready to send an X and Y coordinate, and the subroutine returns with the variable SENSE equal to zero. Otherwise the subroutine continues to wait for you to touch the pad. If the counter C reaches the value 50, then PowerPad is not talking to your computer and the subroutine jumps to the initialization subroutine (INIT PAD).

The fourth subroutine (lines 13000-13070) reads the values of the X and Y coordinate from PowerPad. It is called after the third subroutine has determined that you have touched the pad.

The fourth subroutine calls the fifth subroutine twice: first to read in Y, and again to read in X. The fourth subroutine then tells the pad to resume scanning for another touch. It returns with the variables X and Y set to the position of the point of touch.

The fifth subroutine (lines 14000-14110) reads the value of one number (either X or Y) from PowerPad. It is called only by the fourth subroutine. It is this subroutine which assembles the binary bits coming from PowerPad back into a whole number. The value of the number read is returned in the variable BYTE.

A flow chart (Figure 2) which depicts schematically the BASIC PowerPad Test program follows the listing of Program 1.

Suggestions:

Chalk Board has taken great care to insure that this and subsequent programs have been printed correctly and that they run properly. Should you have difficulty in running any of the programs in this manual, check your entries to be sure that each line has been typed correctly. Once you have typed your program and are sure that it runs properly, we suggest that you save the program on a diskette or cassette to simplify your future use of the program. Refer to the manual which came with your Atari computer for instructions concerning saving programs.

To help you keep track of the programs you create, three PowerLog™ cards are included in this package. Be sure to label your cassette tapes or diskettes carefully and clearly. Use the PowerLog cards to record the identity and location of your own programs.

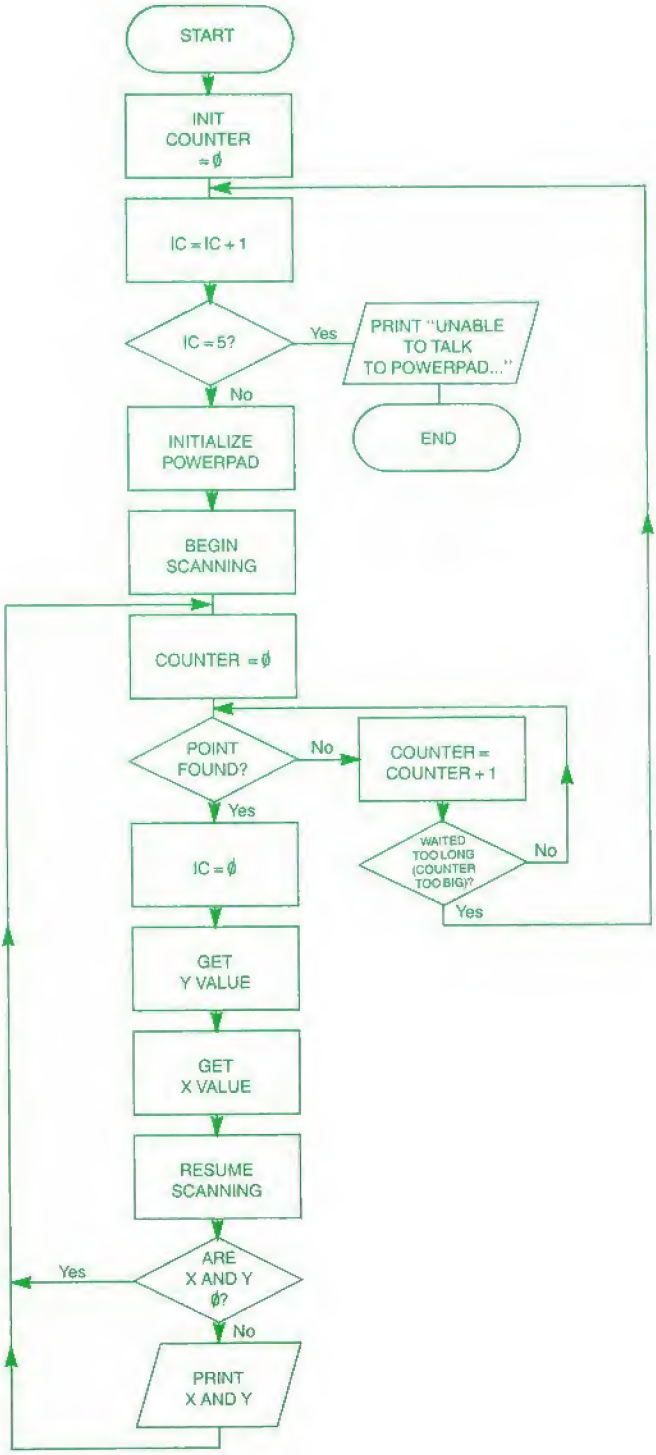
As you type program 1, as well as all of the other sample programs in this manual, you can save time by leaving out the REM statements.

Program 1:
PowerPad Test (all BASIC).

```
1 REM ******* POWERPAD TEST ********
2 REM POWERPAD DEMONSTRATION PROGRAM
3 REM IN BASIC FOR THE ATARI 400/800
4 REM NOTE: USE THE SUBROUTINES IN
5 REM -      LINES 10000-14110
6 REM -      FOR YOUR OWN PROGRAMS
7 REM - (YOU CAN LEAVE OUT ALL REMS)
8 REM CHALK BOARD, INC. 1983
9 REM ****************************
100 ? CHR$(125):REM CLEAR SCREEN
110 ? "POWERPAD TEST"
120 GOSUB 10010:REM INIT VARS
130 GOSUB 11010:REM INIT PAD
140 GOSUB 12010:REM CHECK IF POINT PRESSED
150 GOSUB 13010:REM GET X AND Y
200 REM USE X AND Y
210 IF X=0 AND Y=0 THEN 140
220 ? "X=";X;"  Y=";Y
230 GOTO 140
240 REM ****************************
10000 REM INIT VARIABLES
10010 PACTL=54018:REM $D302 = PIA PACTL
10020 DDRA=54016:REM .$D300 = PIA DDRA
10030 PORTA=54016:REM $D300 = PIA PORTA
10040 IC=0:REM COUNTS # OF CALLS OF INIT PAD
10050 RETURN
10060 REM ***************************
11000 REM INIT PAD
11010 IC=IC+1:IF IC=5 THEN 11110
11020 POKE PACTL,56:REM TO ACCESS DDRA
11030 POKE DDRA,6:REM SET ATARI PORT TO TALK TO PAD
11040 POKE PACTL,60:REM TO ACCESS PORTA
11050 POKE PORTA,0:REM ZERO CLOCK AND CLEAR
11060 POKE PORTA,2:REM PULSE CLEAR LINE
11070 POKE PORTA,0:REM (BEGIN SCANNING)
11080 RETURN
```

```
11090 REM ==========
11100 REM INIT HAS BEEN CALLED 5 TIMES
11110 ? "UNABLE TO TALK TO POWERPAD.  DID"
11120 ? "YOU PLUG PAD INTO JOYSTICK PORT #1?"
11130 END
11140 REM ***************************
12000 REM CHECK IF POINT PRESSED
12010 C=0:REM COUNTER
12020 SENSE=PEEK(PORTA):REM CHECK SENSE LINE
12030 K=(INT(SENSE/16))*16:REM REMOVE ALL BITS
12040 SENSE=SENSE-K:REM - EXCEPT BIT 3
12050 SENSE=INT(SENSE/8):REM NOW EITHER 0 OR 1
12060 IF SENSE=1 THEN 12080:REM POINT NOT FOUND
12070 IC=0:RETURN :REM POINT FOUND
12080 C=C+1:IF C<50 THEN 12020:REM 50 TRIES
12090 GOSUB 11010:REM CALL INIT AGAIN
12100 GOTO 12010:REM CHECK SENSE AGAIN
12110 REM ***************************
13000 REM GET X AND Y
13010 POKE PORTA,4:REM PULSE CLOCK LINE
13020 POKE PORTA,0:REM (SHIFT OFF FIRST BIT)
13030 GOSUB 14010:Y=BYTE:REM GETBYTE
13040 GOSUB 14010:X=BYTE:REM GETBYTE
13050 POKE PORTA,2:REM PULSE CLEAR LINE
13060 POKE PORTA,0:REM (CONTINUE SCANNING)
13070 RETURN
13080 REM ***************************
14000 REM GET BYTE
14010 BYTE=0
14020 FOR L=1 TO 7:REM READ AND COMBINE 7 BITS
14030 POKE PORTA,4:REM PULSE CLOCK LINE
14040 POKE PORTA,0:REM (SHIFT IN NEXT BIT)
14050 DTA=PEEK(PORTA):REM READ DATA LINES
14060 K=(INT(DTA/2))*2:REM REMOVE ALL BUT BIT 0
14070 DTA=DTA-K:REM NOW EITHER 0 OR 1
14080 IF DTA=0 THEN BYTE=BYTE+128:REM BUILD VALUE
14090 BYTE=BYTE/2:REM -                  IN BYTE
14100 NEXT L
14110 RETURN
```

Figure 2. Flowchart of PowerPad Test Program



How PowerPad Works

This section is intended for the experienced programmer who wants to know more about how PowerPad communicates with the Atari computer. The machine language version of the PowerPad Test program appears at the end of this section. You need not understand machine language to take advantage of this faster-running program.

PowerPad's Interface Lines and Shift Register

PowerPad uses four lines to interface to your Atari computer. The CLOCK and CLEAR lines are outputs from the computer to PowerPad. The DATA and SENSE lines are outputs from PowerPad to the computer.

The CLEAR line is used to tell PowerPad to scan for a touch point. It must be pulsed high (brought from low to high and back to low again) to begin the scan.

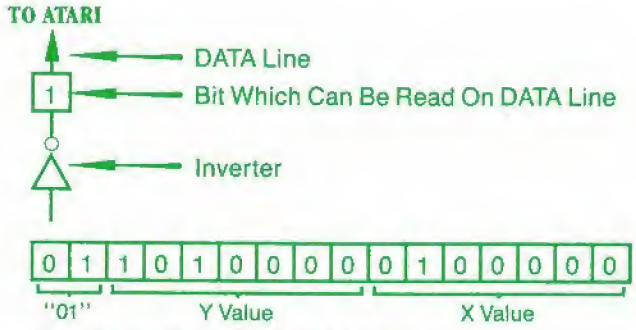
The SENSE line is PowerPad's signal that a touch has occurred. This line is normally high. When you touch the pad, the line goes low. It remains low until the CLEAR line is pulsed.

When PowerPad senses a touch, it stores the X and Y coordinate values in a 16-bit shift register. Figure 3a shows an example of what the shift register might contain immediately after the pad has sensed a touch. The left-most two bits are always loaded with "01." The next seven bits to the right contain the value for Y, and the right-most seven bits hold the value for X. The X and Y values are both stored with their least significant bit to the left. So in Figure 3a, the shift register is shown holding the coordinates (X=2, Y=5).

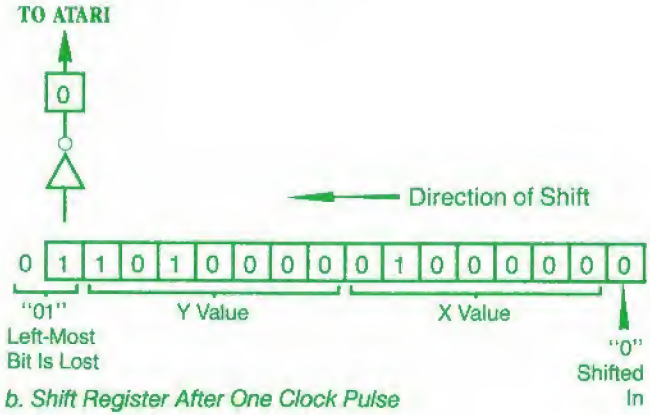
At all times the computer can read only the left-most bit of the shift register. This bit is inverted (a 1 becomes a 0, and a 0 becomes a 1) and then sent out on PowerPad's DATA line.

The CLOCK line enables the computer to read the other bits in the shift register. Each time the CLOCK line is pulsed (set from low to high and back to low again), the shift register moves its contents one position to the left. The left-most bit is lost and a "0" is shifted into the right-most position. Figure 3b shows the contents of the shift register after the CLOCK line has been pulsed once. Figure 3c shows the register after a second shift. The first two shifts are always necessary to remove the "01" in the left-most bit positions. At this point the computer can begin reading the seven bits corresponding to the Y value, followed by the seven bits for the X value. Remember that since the bits coming out of the DATA line are inverted, the computer must invert them again.

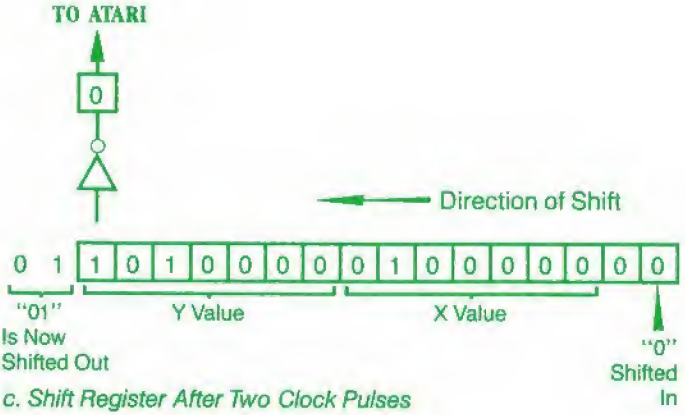
The CLOCK line should be held low while PowerPad is scanning. If the line is pulsed before the pad finds a point, the shift register immediately loads garbage while the pad continues to scan for a point. Then when the pad finds the next point, it is unable to put the correct data into the shift register. At this point, pulsing the CLEAR line clears the garbage from the register and causes the pad to resume normal operation.



a. Initial Contents of Shift Register



b. Shift Register After One Clock Pulse



c. Shift Register After Two Clock Pulses

Figure 3. Shift Register

Programming Considerations

As PowerPad scans its matrix of switches, it starts at coordinates (X=0, Y=0), then looks at (X=0, Y=1), etc., up to (X=0, Y=119). It then looks at the next column, (X=1, Y=0) up to (X=1, Y=119), and continues scanning each successive column until it reaches (X=119, Y=119). At this point PowerPad scans again, starting with (X=0, Y=0). If more than one point is closed in the switch matrix, PowerPad scans to the first point, lowers the SENSE line and waits until the computer reads the coordinates. When the computer acknowledges reading the coordinates (by pulsing the CLEAR line), PowerPad resumes scanning the switch matrix at the next point following the one it just reported. When it finds another closed switch in the matrix, it again lowers the SENSE line and waits for the computer to read the coordinates before continuing to scan.

Each time PowerPad scans through the point (X=0, Y=0), it reports a switch closure at that point, even if nothing is touching the pad. This is a very useful feature. If a program reads the pad and finds coordinates (X=0, Y=0) two times in succession, then it knows that PowerPad has scanned all the way through its switch matrix without finding any closed switch points. In other words, this indicates that nothing is touching the pad. Programs can use this feature to detect when you have lifted your finger off PowerPad.

It is possible that some points in PowerPad's switch matrix may close permanently due to extraordinary wear and tear. When a point becomes shorted, PowerPad reports its coordinates each time it scans through the matrix, just as if you were applying continuous pressure to that point. Writing a routine to detect shorted points and to ignore them is not difficult. At the start of a program, create a table of all shorted points (points reported as being touched while nothing is really touching PowerPad) with the exception of (X=0, Y=0). Later in the pro-

gram, whenever a location read from the pad matches any of the points in the table, the program ignores that location.

Atari's Joystick Port Interface

When you plug PowerPad into an Atari joystick port, the CLOCK, CLEAR, SENSE, and DATA lines from the PowerPad are connected to Port A of the 6520 Peripheral Interface Adapter chip (PIA) inside the Atari. (See Figure 4.) All PowerPad software expects to find PowerPad in joystick port 1.

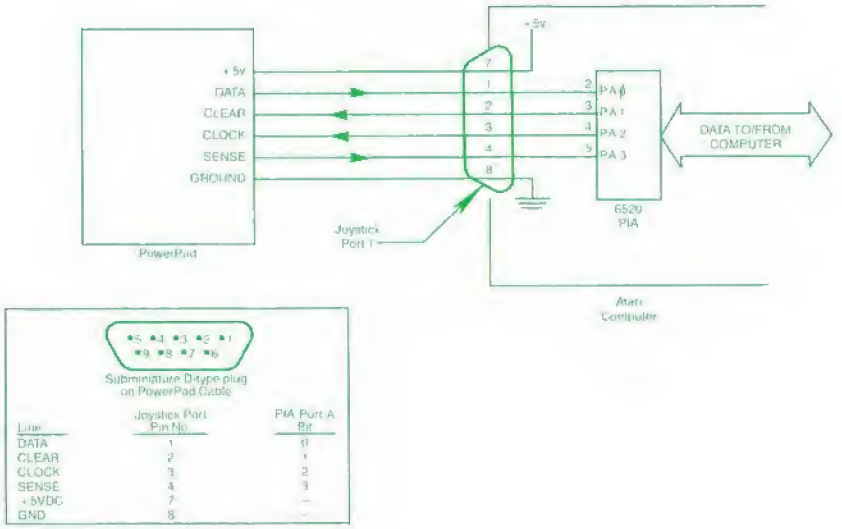


Figure 4. Atari Joystick Interface

The registers of the 6520 PIA are memory mapped in the Atari to locations $D300 (decimal 54016) through $D303 (decimal 54019). You can access the registers by loading and storing (or PEEKing and POKEing) data at these addresses. There are three particular registers in the 6520 with which we are concerned. They are: the Port A Control Register, or PACTL, located at $D302; the Port A Data Direction Register, or PADDR, at $D300; and the Port A Register, or PORTA, also at $D300.

Although PADDR and PORTA both share the same location, only one is accessible at a time. Bit 2 of PACTL determines which of the two is active. When bit 2 is set to 0, $D300 accesses PADDR, and when bit 2 is set to 1, $D300 accesses PORTA. To set up communication with PowerPad, you will first need to access PADDR.

The individual bits of PADDR determine whether a line on the joystick port will be an input to or an output from the Atari. Setting a bit to 0 assigns the corresponding line as an input, and setting it to 1 assigns the line as an output. PowerPad requires that lines 0 and 3 be inputs, and lines 1 and 2 be outputs. So you should store the value 6 (binary 0110) as the low four bits of PADDR.

Once you have set up PADDR you can read the DATA and SENSE lines and write to the CLOCK and CLEAR lines through PORTA. Setting a bit of PORTA to 0 or 1 sets the corresponding output line low or high, respectively. Setting an input line low or high sets its corresponding PORTA bit to 0 or 1, respectively. Given below are machine language examples of how to set up and talk to PowerPad.

```
LDA $D302      ;  get current value of PACTL
AND #$FB       ;  set bit 2 to 0
STA $D302      ;  make PADDR accessible at $D300

LDA #$06
STA $D300      ;  set up PADDR (2 inputs & 2 outputs)

LDA $D302      ;  get current value of PACTL
ORA #$04       ;  set bit 2 to 1
STA $D302      ;  make PORTA accessible at $D300

LDA #$00
STA $D300      ;  set CLEAR and CLOCK low

LDA #$02
STA $D300      ;  set CLEAR high and CLOCK low

LDA #$04
STA $D300      ;  set CLOCK high and CLEAR low

LDA $D300      ;  read the DATA line
AND $#01       ;  (A = 0 for DATA = low
               ;   A = 1 for DATA = high)

LDA $D300      ;  read the SENSE line
AND $#08       ;  (A = 0 for SENSE = low
               ;   A = 8 for SENSE = high)
```

While entering Program 2, make certain that the DATA statements are typed **exactly** as shown here. Otherwise the program will not run properly and could even cause the computer to "freeze." If you have a disk or cassette, save the program before running it.

These Data statements contain the machine language subroutines taken from the listing in Appendix B.

Program 2:
PowerPad Test (BASIC with machine language subroutines. See Appendix B for an assembly language listing of the subroutines).

```
1 REM ******** POWERPAD TEST ********
2 REM POWERPAD DEMONSTRATION PROGRAM
3 REM IN BASIC WITH MACHINE LANGUAGE
4 REM SUBROUTINES FOR ATARI 400/800.
5 REM NOTE: USE THE SUBROUTINES IN
6 REM -      LINES 10000-15370
7 REM -      FOR YOUR OWN PROGRAMS
8 REM - (YOU CAN LEAVE OUT ALL REMS)
9 REM CHALK BOARD, INC. 1983
10 REM ******************************
100 ? CHR$(125):REM CLEAR SCREEN
110 ? "POWERPAD TEST"
120 GOSUB 10010:REM SET UP M.L.
130 GOSUB 11010:REM CALL INITPAD
140 GOSUB 12010:REM CALL GETSENSE
150 GOSUB 13010:REM CALL GETX AND GETY
200 REM USE X AND Y
210 IF X=0 AND Y=0 THEN 140
220 ? "X=";X;"  Y=";Y
230 GOTO 140
240 REM ******************************
10000 REM INSTALL MACH. LANG. SUBROUTINES
10010 ? "PLEASE WAIT";
10020 DIM BYTE$(2),IP$(45),GS$(28),GY$(56),GX$(54)
10030 IP=ADR(IP$):REM SET NUMERIC VARIABLE
10040 GS=ADR(GS$):REM TO ADDRESS OF FIRST
10050 GY=ADR(GY$):REM CHARACTER IN THE
10060 GX=ADR(GX$):REM STRING.
10070 ADDR=IP:N=45:GOSUB 14010
10080 ADDR=GS:N=28:GOSUB 14010
10090 ADDR=GY:N=52:GOSUB 14010
10100 ADDR=GX:N=52:GOSUB 14010
10110 IC=0:REM COUNTS CALLS OF INIT
10120 ? "DONE INSTALLING M.L."
10130 RETURN
10140 REM ADDR IS STARTING ADDRESS OF
10150 REM SUBROUTINE.  N IS NUMBER OF
10160 REM BYTES LONG.
10170 REM ***************************
11000 REM CALL INIT PAD
11010 IC=IC+1:IF IC=5 THEN 11110
11020 L=USR(IP)
11030 RETURN
11040 REM ==========
11100 REM INIT PAD HAS BEEN CALLED 5 TIMES
11110 ? "UNABLE TO TALK TO POWERPAD.  DID"
11120 ? "YOU PLUG PAD INTO JOYSTICK PORT 1?"
11130 END
11140 REM ***************************
```

```
12000 REM CALL GETSENSE
12010 SENSE=USR(GS)
12020 IF SENSE=1 THEN 12040:REM POINT NOT FOUND
12030 IC=0:RETURN :REM RESET INIT COUNTER
12040 GOSUB 11010:REM CALL INIT AGAIN
12050 GOTO 12010
12060 REM ***************************
13000 REM CALL GETX AND GETY
13010 Y=USR(GY)
13020 X=USR(GX)
13030 RETURN
13040 REM ***************************
14000 REM STORE SUBROUTINE INTO STRING
14010 ? ".";
14020 FOR L=ADDR TO ADDR+N-1
14030 READ BYTE$
14040 K=ASC(BYTE$(1,1))-48
14050 IF K>9 THEN K=K-7
14060 BYTE=K*16
14070 K=ASC(BYTE$(2,2))-48
14080 IF K>9 THEN K=K-7
14090 BYTE=BYTE+K
14100 POKE L,BYTE
14110 NEXT L
14120 RETURN
14130 REM READ HEXADECIMAL BYTE FROM DATA
14140 REM STATEMENTS; CONVERT BYTE TO
14150 REM DECIMAL; STORE VALUE IN MEMORY
14160 REM ***************************
15000 REM INITPAD (45 BYTES)
15010 DATA 68,AD,02,D3,29,FB,8D,02
15020 DATA D3,A9,06,8D,00,D3,AD,02
15030 DATA D3,09,04,8D,02,D3,A9,00
15040 DATA 8D,00,D3,EA,EA,A9,02,8D
15050 DATA 00,D3,A2,14,CA,D0,FD,8E
15060 DATA 00,D3,EA,EA,60
15100 REM GETSENSE (28 BYTES)
15110 DATA 68,A2,FF,A0,FF,AD,00,D3
15120 DATA 29,08,F0,09,CA,D0,F6,88
15130 DATA D0,F3,4A,4A,4A,85,D4,A9
15140 DATA 00,85,D5,60
15200 REM GETY (52 BYTES)
15210 DATA 68,A9,04,8D,00,D3,A2,14
15220 DATA CA,D0,FD,8E,00,D3,EA,EA
15230 DATA A0,07,A9,04,8D,00,D3,A2
15240 DATA 14,CA,D0,FD,8E,00,D3,EA
15250 DATA EA,AD,00,D3,4A,66,D4,88
15260 DATA D0,E8,A5,D4,49,FF,4A,85
15270 DATA D4,84,D5,60
15300 REM GETX (52 BYTES)
15310 DATA 68,A0,07,A9,04,8D,00,D3
15320 DATA A2,14,CA,D0,FD,8E,00,D3
15330 DATA EA,EA,AD,00,D3,4A,66,D4
15340 DATA 88,D0,E8,A5,D4,49,FF,4A
15350 DATA 85,D4,84,D5,A9,02,8D,00
15360 DATA D3,A2,14,CA,D0,FD,8E,00
15370 DATA D3,EA,EA,60
```

Creating Your Own Command Buttons

A command button is an area on the PowerPad surface which, when pushed, calls a subroutine within your program into action. You can design your own command buttons anywhere on the surface of PowerPad. Program 3 assists you by providing you with the range of values for X and Y corresponding to your command button. You can use the information Program 3 provides to determine when your program will call its command button subroutines.

```
Program 3:
Command Button Mapper.  You must also enter
the machine language subroutines from Program 2
(lines 10000-15370)

1 REM **** COMMAND BUTTON MAPPER ****
2 REM GIVES YOU THE RANGE OF X AND Y
3 REM VALUES FOR YOUR COMMAND BUTTONS
4 REM !REQUIRES THE MACHINE LANGUAGE
5 REM SUBROUTINES (LINES 10000-15370)
6 REM FROM PROGRAM 2!
7 REM (YOU CAN LEAVE OUT ALL REMS)
8 REM CHALK BOARD, INC. 1983
9 REM ******************************
100 ? CHR$(125):REM CLEAR SCREEN
110 GOSUB 1010:REM INSTRUCTIONS & INIT VARS
130 GOSUB 11010:REM CALL INITPAD
140 GOSUB 12010:REM CALL GETSENSE
150 GOSUB 13010:REM CALL GETX AND GETY
200 REM USE X AND Y
210 IF X+Y=0 THEN 140
220 GOSUB 2010:REM UPDATE RANGE
230 GOTO 140
240 REM ******************************
1000 REM INSTRUCTIONS & INIT VARIABLES
1010 XHI=0:REM . LEFTMOST VALUE
1020 XLO=119:REM RIGHTMOST VALUE
1030 YHI=0:REM . BOTTOM VALUE
1040 YLO=119:REM TOP VALUE
1100 ? "COMMAND BUTTON MAPPER":?
1110 ? "TO USE THIS PROGRAM:":?
```

- Insert the grid overlay onto the surface of PowerPad.
- Choose an area of PowerPad's surface to represent your button.
- Use your marking pen to draw a square to help you identify the location of your button. You can label or color your button any way you choose.
- If you need to change a button's location, refer to Appendix A for instructions concerning erasing marks on your overlay.
- Enter Program 3 into your computer.
- RUN the program.

```
1120 ? "~ MOVE YOUR FINGER OVER THE ENTIRE"
1130 ? "  AREA OF YOUR BUTTON, STAYING"
1140 ? "  *INSIDE* ITS OUTLINE."
1150 ? "~ THE MAXIMUM AND MINIMUM X AND Y"
1160 ? "  VALUES OF YOUR BUTTON WILL BE"
1170 ? "  CONTINUOUSLY PRINTED."
1180 ? "~ KEEP PRESSING YOUR BUTTON UNTIL"
1190 ? "  THE X AND Y VALUES STOP CHANGING."
1200 ? "  (THESE ARE THE VALUES YOU SHOULD"
1210 ? "  USE IN YOUR OWN PROGRAM)."
1220 ?
1300 GOSUB 10010:REM SET UP M.L.
1310 ? :? "PRESS ANY KEY WHEN READY"
1320 POKE 764,255:REM CLEAR KEYBOARD
1330 IF PEEK(764)=255 THEN 1330:REM WAIT FOR KEY
1340 POKE 764,255:REM CLEAR KEY
1350 ? CHR$(125)
1360 RETURN
1370 REM ****************************
2000 REM UPDATE & PRINT RANGE OF X & Y
2010 IF X<XLO THEN XLO=X
2020 IF X>XHI THEN XHI=X
2030 IF Y<YLO THEN YLO=Y
2040 IF Y>YHI THEN YHI=Y
2050 ? CHR$(125)
2060 ? "X>";XLO;" AND X<";XHI;
2070 ? " AND Y>";YLO;" AND Y<";YHI
2080 ? :? :?
2090 RETURN
2100 REM ****************************
```

Once you have determined the range of values for X and Y (for example, X might range between 20 and 30, and Y between 40 and 50), you can set up your program to know when you are pressing your command button. Assume that you want your command button to activate a routine which begins at line 3000 in your own program. To call the routine you must insert a line in your original PowerPad Test program to say the following: if I touch PowerPad at this spot, then go to line 3000 and do the routine there. We can represent this in BASIC in the following way:

```
225  IF X>20 AND X<30 AND Y>40 AND Y<50 THEN GOTO 3000
```

In this example **line 3000** is the beginning of the routine you want executed whenever your command button is pressed.

When the program returns from the routine and reads the pad again, it might find that you are still pressing the command button. If the routine is called each time the pad reports a switch closure within your pushbutton area, then the routine is likely to be run several times for each push of the button.

If you do not want the routine running several times for each push of the button, there is an easy way to avoid re-running the routine. Use a variable to indicate whether or not the button has been pressed. When the pad reports a point within your button area, your program checks the variable to see if you have already pressed the button. If not, then it calls the routine to perform the function associated with that button and sets the variable to show that that function has been used.

If the variable shows that you previously pressed the button, then the program ignores that button until you either lift your finger from the pad or touch another button. In both cases the program can then reset the variable so that the button will work the next time you press it. When your program reads the pad and finds two (X=0, Y=0) points in succession, it knows that nothing is touching the pad.

Program 4, Making Change, uses this method to insure that when you hold down a coin button, the value of that coin is added to your total only once.

Sample Programs

Program 4 is an educational game in which you make change for random amounts of money (1–100 cents). Try changing some of the PRINT statements to have fun with the responses.

Creating Your Overlay For Making Change

- Fit the overlay on PowerPad's surface.
- Use a marking pen to divide the overlay into four equal vertical columns. See Figure 5.
- Mark each section with a drawing of its appropriate coin value (left to right: Quarter, Dime, Nickel, Penny).

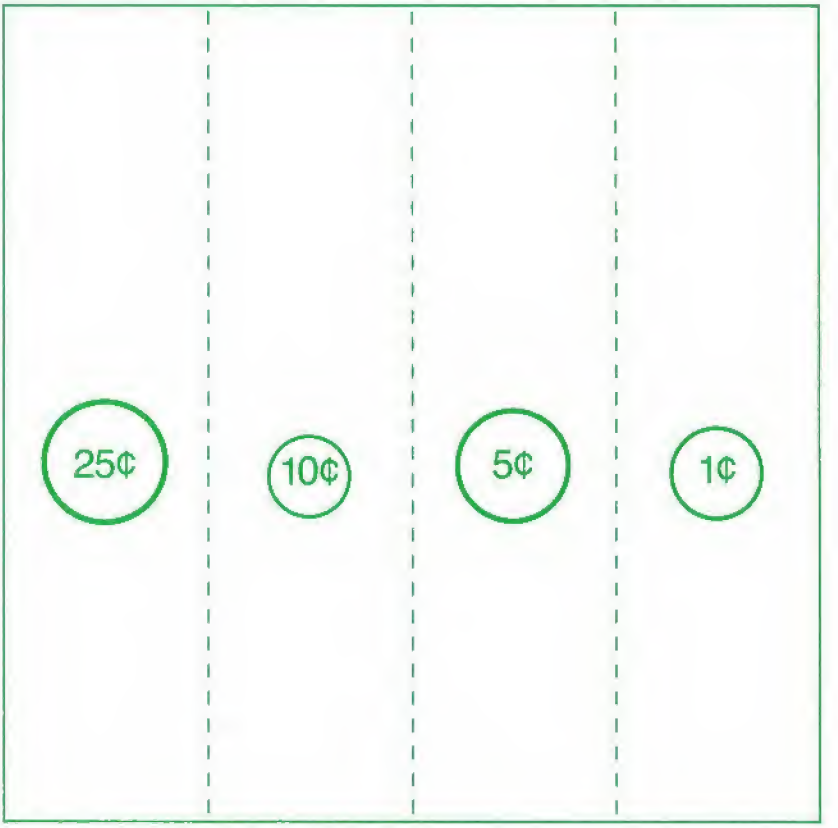


Fig. 5. Overlay For Making Change Program

Program 4:
Making Change. You must also enter the machine language subroutines from Program 2 (lines 10000-15370)

```
1 REM ******** MAKING CHANGE ********
2 REM AN EDUCATIONAL GAME USING THE
3 REM POWERPAD
4 REM !REQUIRES THE MACHINE LANGUAGE
5 REM SUBROUTINES (LINES 10000-15370)
6 REM FROM PROGRAM 2!
7 REM (YOU CAN LEAVE OUT ALL REMS)
8 REM CHALK BOARD, INC. 1983
9 REM ******************************
100 ? CHR$(125):REM CLEAR SCREEN
110 GOSUB 10010:REM SET UP M.L.
120 GOSUB 1010:REM INIT VARS & PICK AMOUNT
130 GOSUB 11010:REM CALL INITPAD
140 GOSUB 12010:REM CALL GETSENSE
150 GOSUB 13010:REM CALL GETX AND GETY
200 REM USE X AND Y
210 GOSUB 2010
220 GOTO 140
230 REM ******************************
1000 REM INIT VARIABLES
1010 PS=0:NS=0:DS=0:QS=0:REM # OF COINS
1020 TTL=0:REM TOTAL VALUE OF COINS
1030 AMT=INT(RND(0)*99)+1:REM # TO MATCH
1040 PF=1:NF=1:DF=1:QF=1:REM "TOUCHED" FLAGS
1050 ?
1060 ? "=====================================":?
1070 ? "LET'S MAKE CHANGE FOR ";AMT;" CENTS":?
1080 RETURN
1090 REM ******************************
2000 REM CONVERT X AND Y TO A COIN
2010 IF X+Y=0 THEN 2100:REM NONE TOUCHED?
2020 ZF=0:REM POINT WAS NOT (0,0)
2030 IF X>=0 AND X<30 THEN 5010:REM PENNY
2040 IF X>=30 AND X<60 THEN 6010:REM NICKEL
2050 IF X>=60 AND X<90 THEN 7010:REM DIME
2060 IF X>=90 AND X<120 THEN 8010:REM QUARTER
2100 ZF=ZF+1:REM COUNT CONSECUTIVE (0,0)'S
2110 IF ZF<2 THEN RETURN :REM ONLY 1 SO FAR
2120 GOSUB 3010:REM 2 SO NOTHING TOUCHING PAD
2130 RETURN
2140 REM ******************************
3000 REM CLEAR ALL "BUTTON TOUCHED" FLAGS
3010 PF=0:NF=0:DF=0:QF=0
3020 RETURN
3030 REM ******************************
```

```
4000 REM SHOW CURRENT TOTAL
4010 ?
4020 ? "MAKING CHANGE FOR ";AMT;" CENTS":?
4030 IF PS>0 THEN ? "    PENNIES: ";PS
4040 IF NS>0 THEN ? "    NICKELS: ";NS
4050 IF DS>0 THEN ? "      DIMES: ";DS
4060 IF QS>0 THEN ? "   QUARTERS: ";QS
4070 ?
4080 ? "YOUR TOTAL SO FAR IS ";TTL;" CENTS":?
4090 IF TTL>=AMT THEN 4110
4100 RETURN :REM NOT ENOUGH YET
4110 IF TTL>AMT THEN 4140
4120 ? "CONGRATULATIONS! YOU GOT IT!"
4130 GOTO 4150
4140 ? "WHOOPS! THAT WAS TOO MUCH."
4150 GOSUB 1010:REM DO IT AGAIN
4160 RETURN
4170 REM ***************************
5000 REM PENNY WAS TOUCHED
5010 IF PF=1 THEN RETURN :REM WAS TOUCHED BEFORE
5020 GOSUB 3010:REM ALLOW ALL OTHER COINS
5030 PF=1:REM SHOW THAT PENNY TOUCHED
5040 PS=PS+1:REM INCREMENT PENNY COUNTER
5050 TTL=TTL+1:REM UPDATE TOTAL
5060 GOTO 4010:REM SHOW TOTAL
5070 REM ***************************
6000 REM NICKEL WAS TOUCHED
6010 IF NF=1 THEN RETURN :REM WAS TOUCHED BEFORE
6020 GOSUB 3010:REM ALLOW ALL OTHER COINS
6030 NF=1:REM SHOW THAT NICKEL TOUCHED
6040 NS=NS+1:REM INCREMENT NICKEL COUNTER
6050 TTL=TTL+5:REM UPDATE TOTAL
6060 GOTO 4010:REM SHOW TOTAL
6070 REM ***************************
7000 REM DIME WAS TOUCHED
7010 IF DF=1 THEN RETURN :REM WAS TOUCHED BEFORE
7020 GOSUB 3010:REM ALLOW ALL OTHER COINS
7030 DF=1:REM SHOW THAT DIME TOUCHED
7040 DS=DS+1:REM INCREMENT DIME COUNTER
7050 TTL=TTL+10:REM UPDATE TOTAL
7060 GOTO 4010:REM SHOW TOTAL
7070 REM ***************************
8000 REM QUARTER WAS TOUCHED
8010 IF QF=1 THEN RETURN :REM WAS TOUCHED BEFORE
8020 GOSUB 3010:REM ALLOW ALL OTHER COINS
8030 QF=1:REM SHOW THAT QUARTER TOUCHED
8040 QS=QS+1:REM INCREMENT QUARTER COUNTER
8050 TTL=TTL+25:REM UPDATE TOTAL
8060 GOTO 4010:REM SHOW TOTAL
8070 REM ***************************
```

Program 5 is a small sample of the graphics capabilities of the Atari in combination with your PowerPad. The program draws a waving man on the screen when you touch the pad.

```
Program 5:
Waving Man.  You must also enter the
machine language subroutines from
Program 2 (lines 10000-15370)

1 REM ********* WAVING MAN *********
2 REM POWERPAD DEMONSTRATION PROGRAM
3 REM !REQUIRES THE MACHINE LANGUAGE
4 REM SUBROUTINES (LINES 10000-15370)
5 REM FROM PROGRAM 2!
6 REM - (YOU CAN LEAVE OUT ALL REMS)
7 REM CHALK BOARD, INC. 1983
8 REM ******************************
100 ? CHR$(125):REM CLEAR SCREEN
110 GOSUB 1010:REM SET UP GRAPHICS CHARS
120 GOSUB 10010:REM SET UP M.L.
130 GOSUB 11010:REM CALL INITPAD
140 GOSUB 12010:REM CALL GETSENSE
150 GOSUB 13010:REM CALL GETX AND GETY
200 REM USE X AND Y
210 IF X+Y=0 THEN 140
220 GOSUB 2010
230 GOTO 140
240 REM ******************************
1000 REM INIT WAVING MAN
1010 DIM HD$(3),TP$(3),MD$(3),LG$(3),TL$(3),TR$(3)
1020 HD$(1)=" "
1030 HD$(2)=CHR$(20)
1040 HD$(3)=" "
1050 TP$(1)=CHR$(18)
1060 TP$(2)=CHR$(19)
1070 TP$(3)=CHR$(18)
1080 MD$(1)=" "
1090 MD$(2)=CHR$(124)
1100 MD$(3)=" "
1110 LG$(1)=CHR$(6)
1120 LG$(2)=" "
1130 LG$(3)=CHR$(7)
1140 TL$=TP$:TL$(1)=CHR$(26)
1150 TR$=TP$:TR$(3)=CHR$(3)
1160 GRAPHICS 0:REM INIT SCREEN
1170 POKE 752,1:REM MAKE CURSOR INVISIBLE
1180 RETURN
1190 REM ****************************
```

```
2000 REM DISPLAY WAVING MAN
2010 XP=(((119-X)*36)/119)+1
2020 YP=((Y*17)/119)+1
2030 ? CHR$(125)
2040 POSITION XP,YP:REM DRAW HEAD
2050 ? HD$
2060 POSITION XP,YP+1:REM DRAW ARMS
2070 ? TP$
2080 POSITION XP,YP+2:REM DRAW WAIST
2090 ? MD$
2100 POSITION XP,YP+3:REM DRAW LEGS
2110 ? LG$
2120 FOR L=1 TO 5:REM WAVE
2130 FOR D=1 TO 15:NEXT D:REM DELAY
2140 POSITION XP,YP+1
2150 IF X>60 THEN ? TL$
2160 IF X<61 THEN ? TR$
2170 FOR D=1 TO 15:NEXT D:REM DELAY
2180 POSITION XP,YP+1
2190 ? TP$
2200 NEXT L
2210 RETURN
2220 REM ****************************
```

Questions to Ponder

A. How would you modify Making Change so that you must touch the pad only within the outline of each coin?

B. How could you change the PowerPad Test program to print out the value (1 or 0) for each data bit read in from the pad? To print the state of the SENSE line (high or low)?

C. On your PowerPad overlay draw a medium-sized rectangle with sides parallel to the sides of PowerPad. Then RUN the Waving Man program (Program 5). Next touch any three of the four corners of your rectangle simultaneously. Continue touching the three corners until the waving man has appeared at all positions several times. What do you notice about the route of the waving man?

The waving man appears at the fourth corner of the rectangle—even though no pressure was applied to that point. This "false-image" is

also known as "aliasing." What would explain this occurrence? How can you avoid this in your own programs? How can you take advantage of this in your own programs?

Do you have any ideas that you would like to share with us about your experiences with the PowerPad Programming Kit? Do you want to know more about what other people are doing with PowerPad and Chalk Board products? If so, please write to us at the address given at the front of this booklet.

Your name will be placed on a list of users who receive our company newsletter. If you send us some of your ideas, you might appear in an upcoming issue.

Watch for additions to Leonardo's Library of quality software for your home computer.

Appendix A. Care of Overlays, Replacement Overlays

The overlay is designed for use with colored marking pens of the type included in the Programming Kit. Marks made by these pens should not smudge or smear significantly with normal use. The use of colored marking pens of another type is not recommended because they can smear or be more difficult to erase.

To clean an overlay, use a damp cloth or paper towel. **Always** remove the overlay from PowerPad before cleaning because water may damage the surface of PowerPad.

Additional overlays may be ordered from:

Customer Support
Chalk Board, Inc.
Suite 140
3772 Pleasantdale Rd.
Atlanta, Georgia 30340

The charge for each replacement overlay is $6.00, which includes $3.00 for the overlay itself and $3.00 for postage and handling.

Appendix B. Machine Language Subroutines Used By Programs 2, 3, 4 and 5

```
1  ;*************************************************
2  ;  Machine language subroutines to interface
3  ;       PowerPad to the Atari 400/800.
4  ;          Chalk Board, Inc. 1983
5  ;*************************************************
6
7  ; Define constants:
8
9  PACTL   =  $D302      ;PIA Port A Control Register
10 PADDR   =  $D300      ;PIA Port A Data Dir. Reg.
11 PORTA   =  $D300      ;PIA Port A Register
12
13 CLRHIGH =  $02        ;to pulse CLEAR     (0000 0010)
14 CLKHIGH =  $04        ;to pulse CLOCK     (0000 0100)
15 SNSMASK =  $08        ;mask for SENSE bit (0000 1000)
16
17 INTADDR =  $06        ;to init Port A DDR (0000 0110)
18
19 RETVAL  =  $D4        ;page 0 location for value to
20                       ;  be returned by BASIC USR
21
22 ; The following delays are necessary because of the
23 ; control port interface:
24
25 ; CLEAR and CLOCK outputs-
26 ;   50 usec to go from low to high
27 ;    1 usec to go from high to low
28
29
```

```
                        30 ; DATA and SENSE inputs-
                        31 ;     2 usec to go from low to high
                        32 ;     2 usec to go from high to low
                        33
                        34 ; processor speed = 1 usec per cycle
                        35
0014                    36 DELVAL  =   $14          ;used by delay loops
                        37                          ;giving 102 cycle delay


                        38 ;************************************************
                        39 ;Subroutine INITPAD
                        40
                        41 ;Initialize the Atari PIA as follows:
                        42
                        43 ; Port A     Control       Dir.      Line From
                        44 ;  bit      Port 1 pin                PowerPad
                        45 ;   0           1          input        DATA
                        46 ;   1           2          output       CLEAR
                        47 ;   2           3          output       CLOCK
                        48 ;   3           4          input        SENSE
                        49
                        50 ;Initialize PowerPad:
                        51
                        52 ;   Reset CLEAR and CLOCK lines to 0
                        53 ;   Pulse CLEAR line to begin scanning
                        54 ;************************************************
0000                    55         .=  0
0000                    56 INITPAD:
0000 68                 57         PLA              ;(pushed by USR function;
                        58                          ;   can be thrown away)
0001 AD 02 D3           59         LDA PACTL        ;get current value;
```

```
0004 29 FB      60          AND #$FB        ;   set bit 2 to 0 so that
0006 8D 02 D3   61          STA PACTL       ;   PADDR is accessible
0009 A9 06      62          LDA #INTADDR    ;set up Port A bits 1 and 2 as
000B 8D 00 D3   63          STA PADDR       ;   outputs, 0 and 3 as inputs
000E AD 02 D3   64          LDA PACTL       ;get current value;
0011 09 04      65          ORA #$04        ;   set bit 2 to 1 so that
0013 8D 02 D3   66          STA PACTL       ;   PORTA is accessible
0016 A9 00      67          LDA #$00        ;reset CLEAR and CLOCK lines
0018 8D 00 D3   68          STA PORTA       ;   to 0
001B EA         69          NOP             ;   wait 4 cycles
001C EA         70          NOP
001D A9 02      71          LDA #CLRHIGH    ;pulse CLEAR line high:
001F 8D 00 D3   72          STA PORTA       ;   (begin scanning)
0022 A2 14      73          LDX #DELVAL     ;   wait
0024            74 IPLOOP:
0024 CA         75          DEX
0025 D0 FD      76          BNE IPLOOP      ;   (count down to 0)
0027 8E 00 D3   77          STX PORTA       ;   bring back low
002A EA         78          NOP             ;   wait 4 cycles
002B EA         79          NOP
002C 60         80          RTS             ;return to BASIC
```

```
                        ;************************************************
                        ;Subroutine GETSENSE

                        ;Examine the SENSE line from the PowerPad
                        ;Return:
                        ;   0 if low (point found), or
                        ;   1 if high (still scanning)
                        ;************************************************
0000                            .=  0
0000                    GETSENSE:
0000 68                         PLA             ;pushed by USR; throw away
0001 A2 FF                      LDX #$FF        ;init loop counters
0003 A0 FF                      LDY #$FF        ;   (do $FFFF tries)
0005                    GSLOOP:
0005 AD 00 D3                   LDA PORTA       ;check SENSE line  (bit 3)
0008 29 08                      AND #SNSMASK    ;   mask out all but bit 3
000A F0 09                      BEQ GSEXIT      ;is it 0?  if so, then done
000C CA                         DEX             ;no, so decrement X loop
000D D0 F6                      BNE GSLOOP      ;is X>0?  if so, branch
000F 88                         DEY             ;no, so decrement Y loop
0010 D0 F3                      BNE GSLOOP      ;is Y>0?  if so, branch
0012 4A                         LSR A           ;no, so fall through
0013 4A                         LSR A           ;   and shift bit 3 down
0014 4A                         LSR A           ;   into bit 0
0015                    GSEXIT:
0015 85 D4                      STA RETVAL      ;save return value (low byte)
0017 A9 00                      LDA #$00
0019 85 D5                      STA RETVAL+1    ;   (high byte = 0)
001B 60                         RTS             ;return to BASIC
```

```
                    110 ;***********************************************
                    111 ;Subroutine GETY
                    112
                    113 ;Strip off the initial two bits ("01"), then read
                    114 ;and assemble the next 7 bits.  Return Y value.
                    115 ;***********************************************
0000                116         .=  0
0000                117 GETY:
0000 68             118         PLA             ;pushed by USR; throw away
0001 A9 04          119         LDA #CLKHIGH    ;pulse CLOCK line (shift off
0003 8D 00 D3       120         STA PORTA       ;   left shift register bit)
0006 A2 14          121         LDX #DELVAL     ;   wait
0008                122 GYLOOPA:
0008 CA             123         DEX
0009 D0 FD          124         BNE GYLOOPA     ;   (count down to 0)
000B 8E 00 D3       125         STX PORTA       ;   bring back low
000E EA             126         NOP             ;   wait 4 cycles
000F EA             127         NOP
0010 A0 07          128         LDY #$07        ;shift in next 7 bits
                    129
0012                130 GYLOOPB:
0012 A9 04          131         LDA #CLKHIGH    ;pulse CLOCK line (shift off
0014 8D 00 D3       132         STA PORTA       ;   left shift register bit)
0017 A2 14          133         LDX #DELVAL     ;   wait
0019                134 GYLOOPC:
0019 CA             135         DEX
001A D0 FD          136         BNE GYLOOPC     ;   (count down to 0)
001C 8E 00 D3       137         STX PORTA       ;   bring back low
001F EA             138         NOP             ;   wait 4 cycles
0020 EA             139         NOP
0021 AD 00 D3       140         LDA PORTA       ;read DATA line (bit 0)
0024 4A             141         LSR A           ;shift bit 0 into carry
0025 66 D4          142         ROR RETVAL      ;shift carry into bit 7
0027 88             143         DEY             ;decrement bit counter
```

```
0028 D0 E8      144         BNE GYLOOPB     ;done 7 yet?  if not, continue
002A A5 D4      145         LDA RETVAL      ;get assembled number
002C 49 FF      146         EOR #$FF        ;invert all of the bits
002E 4A         147         LSR A           ;do 8th shift to right justify
002F 85 D4      148         STA RETVAL      ;save return value (low byte)
0031-84 D5      149         STY RETVAL+1    ;   (high byte = 0)
0033 60         150         RTS             ;return to BASIC


                151 ;***********************************************
                152 ;Subroutine GETX
                153
                154 ;Read and assemble the next 7 bits, resume scan.
                155 ;Return X value.
                156 ;***********************************************
0000            157         .=  0
0000            158 GETX:
0000 68         159         PLA             ;pushed by USR; throw away
0001 A0 07      160         LDY #$07        ;shift in 7 next bits
0003            161 GXLOOPA:
0003 A9 04      162         LDA #CLKHIGH    ;pulse CLOCK line (shift off
0005 8D 00 D3   163         STA PORTA       ;   left shift register bit)
0008 A2 14      164         LDX #DELVAL     ;   wait
000A            165 GXLOOPB:
000A CA         166         DEX
000B D0 FD      167         BNE GXLOOPB     ;   (count down to 0)
000D 8E 00 D3   168         STX PORTA       ;   bring back low
0010 EA         169         NOP             ;   wait 4 cycles
0011 EA         170         NOP
0012 AD 00 D3   171         LDA PORTA       ;read DATA line (bit 0)
0015 4A         172         LSR A           ;shift bit 0 into carry
0016 66 D4      173         ROR RETVAL      ;shift carry into bit 7
0018 88         174         DEY             ;decrement bit counter
```

```
0019 D0 E8      175          BNE GXLOOPA     ;done 7 yet?  if not, continue
001B A5 D4      176          LDA RETVAL      ;get assembled number
001D 49 FF      177          EOR #$FF        ;invert all of the bits
001F 4A         178          LSR A           ;do 8th shift to right justify
0020 85 D4      179          STA RETVAL      ;save return value (low byte)
0022 84 D5      180          STY RETVAL+1    ;   (high byte = 0)
                181
0024 A9 02      182          LDA #CLRHIGH    ;pulse CLEAR line:
0026 8D 00 D3   183          STA PORTA       ;   (resume scanning)
0029 A2 14      184          LDX #DELVAL     ;   wait
002B            185 GXLOOPC:
002B CA         186          DEX
002C D0 FD      187          BNE GXLOOPC     ;   (count down to 0)
002E 8E 00 D3   188          STX PORTA       ;   bring back low
0031 EA         189          NOP             ;   wait 4 cycles
0032 EA         190          NOP
0033 60         191          RTS             ;   return to BASIC
```

Bibliography

Boom, Michael. *Understanding Atari Graphics.* Sherman Oaks, California: Alfred Publishing Company, Inc., 1982.

Carris, Bill. *Inside Atari BASIC.* Reston, Virginia: Reston Publishing Company, Inc., 1983.

Carris, Bill. *Inside Atari Graphics.* Reston, Virginia: Reston Publishing Company, Inc., 1983.

Hayes, David Alan. Control your environment with the Atari 400/800. *Byte Magazine,* July 1983, pp. 428–436.

Inman, Don and Kurt Inman. *The Atari Assembler.* Reston, Virginia: Reston Publishing Company, Inc., 1981.

Leventhal, Lance. *6502 Assembly Language.* Berkeley, California: Osborne/McGraw Hill, 1979.

Papert, Seymour. *Mindstorms.* New York, New York: Basic Books, Inc., 1980.

Poole, Ian, et al. *Your Atari Computer.* Berkeley, California: Osborne/McGraw Hill, 1982.

Views from an Educator

The search is on for ways to use computers to design effective educational environments in homes. In my judgment, Chalk Board leads this important exploration to rediscover the home setting as a powerful force for learning. Simply stated, Chalk Board is rapidly becoming synonymous with learning through home computers. This company is committed to building responsive environments that will help more people reach their full potential.

To this end, Chalk Board has created PowerPad and Leonardo's Library.™ The touch-sensitive pad offers learners a new method of communication with computers. PowerPad is complemented by a library of software that is well conceptualized and carefully written. Each package addresses its unique learning objectives but also remains interrelated with other library units. In the family setting, the software, by its very design, stimulates intelligence, motivation and achievement. As I view it, this integrated system provides significant opportunities for learning not only for those people who have previously experienced academic success, but also for those whose potential has not yet been realized.

Here are four reasons why the Chalk Board PowerPad and its software library serve as an important force for learning:

- People find initial learning experiences meaningful and the desire to repeat them furnishes continuing motivation for learning.
- Learners put forth a real effort and gain satisfaction from accomplishing a task they find challenging.
- Learners discover the pleasure that comes from having acquired and used new understandings, new interests and new skills.

- When teams of people use PowerPad, they develop an interest in working with others for a common purpose.

Further, these conditions for learning take on even more significance for individuals because the PowerPad has built-in flexibility that gives learners personal power to structure the activities of the computer.

The PowerPad Programming Kit

The PowerPad Programming Kit is designed to improve the skill and expand the programming powers of those who are familiar with the BASIC language. This kit allows access to the computer which by-passes the keyboard and enables you to activate lengthy and intricate programs with the simple touch of a finger. The Kit also raises questions that further challenge you to consider new ways to program PowerPad beyond its present applications.

With PowerPad and the PowerPad Programming Kit, you can design your own board and arcade games, adventure games, educational programs, sound and graphics manipulations, and business and home management programs—all of which you operate by touching PowerPad.

The PowerPad Programming Kit contains tools you can use to travel further into your understanding of high technology machinery. With these tools you can develop your powers of analytical thinking, inductive and deductive reasoning, problem solving, and observation. The experiences the Kit offers reinforce your mathematical and scientific training while encouraging the joining of science and your imagination.

Chalk Board, then, has created an exciting way to expand opportunities for learning. PowerPad and Leonardo's Library will provide parents, teachers and other educators with valuable assistance as we enter a period of increased use of home computers as a constructive means for quality education.

Dr. Robert L. Sinclair
Professor of Education and
Director, Center for Curriculum Studies
University of Massachusetts at Amherst

PadMasters Guild

Chalk Board, Inc. is making a special offer to you as an owner of a PowerPad and the PowerPad Programming Kit. You are invited to become a member of the PadMasters Guild. For a full year, your membership is free. After that, the annual membership and renewal fee is $9.95. Your membership includes:

- Quarterly publications containing new programs to run on your PowerPad. These publications serve as a clearing house for programs submitted by programmers like yourself.
- Issues of the Chalk Board newsletter.
- Access to a special "Hot Line" telephone number which you can use for technical assistance.
- Special issues and releases about new product data from Chalk Board, Inc.

If you would like to become a member of the PadMasters Guild, complete this membership form and send it to:

Chalk Board, Inc.
3772 Pleasantdale Road
Atlanta, Georgia 30340

PadMasters Guild

Membership Form

Name _____

Address _____

City _____ State _____ Zip _____

Age _____

Type of Computer You Own/Use _____

Type of Storage Device Used

_____ diskette _____ cassette _____ none

Purpose for which you use a Computer (check all that apply.)

_____ Games _____ Business _____ Education

_____ Word Processing _____ Home Finances

Complete and send this form to: Chalk Board, Inc.
3772 Pleasantdale Road
Atlanta, GA 30340

Product Design:
Ben Satterfield (author, educational applications)
Rod Price (engineer)
Don Higgins (programming engineer)
David Carter (programming engineer)

Support:
Margaret Gorley (educational applications)
Tim Cope (programming engineer)
Kathy Williams (editor)
Margaret Walsh (editor)

Package & User's Guide Design:
Taylor & Taylor, Inc./Atlanta, GA

A touch of genius.